2

AD-A221 553

SAF/AQT-SR-90-006

KASH: A GENERAL PURPOSE KNOWLEDGE ACQUISITION SHELL

Christopher R. Westphal
Duc T. Tran

March 1990

UNITED STATES AIR FORCE
SCIENTIFIC AND TECHNICAL INFORMATION PROGRAM
CONTRIBUTIONS TO INFORMATION SCIENCE

USAF STINFO CONTRIBUTION 90/2

SECRETARY OF THE AIR FORCE
DEPUTY FOR SCIENTIFIC AND TECHNICAL INFORMATION (SAF/AQT)
THE PENTAGON
WASHINGTON, DC 20330-1000

90 05 15 059

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
Mar 90

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
KASH: A General Purpose Knowledge Acquisition Shell.

5. FUNDING NUMBERS

6. AUTHOR(S)
Christopher R. Westphal (IDA)
Duc T. Tran (Consultant)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Secretary of the Air Force
Deputy for Scientific and Technical Information
SAF/AQT, Room 4D289, The Pentagon
Washington, DC 20330-1000

8. PERFORMING ORGANIZATION REPORT NUMBER
USAF-STINFO
Contribution-90/2
SAF/AQT-SR-90-006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Presented at and published in Proceedings of FLAIRS- Florida Artificial Intelligence Research Symposium, Coco Beach, Florida, 4-7 April 1990.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public relears; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
The field of expert systems has numerous and succesful applications ranging from simple tasks to very complex ones. To construct a system requires the developer to isolate a set of rules that will guide the decision making process. Rules are conventionally defined during a series of interviews between a domain expert and a knowledge engineer. The encoding and representation of the extracted domain knowledge has proven to be a difficult barrier to overcome. KASH (Knowledge Acquisition SHell) has been designed to address this problem by providing a knowledge engineer with a set of utilities for constructing knowledge acquisition sessions base on interviewing techniques. The information elicited from domain experts during the sessions is guided by a question dependency graph (QDG). The QDG, defined by the knowledge engineer, is a series of control questions about the domain that are used to organize the cognitive processes of an expert. The content information supplied by the expert, in response to the questions, is represented in in the form of a concept map. These maps can be constructed in a top-down or bottom-up manner by the QDG and used by KASH to generate the rules for a large class of expert system domains.

14. SUBJECT TERMS
Expert systems, Artificial intelligence, knowledge acquisition knowledge engineer, domain expert, KASH, QDG, rules, knowledge base

15. NUMBER OF PAGES
8

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

KASH: A General Purpose Knowledge Acquisition Shell

Christopher R. Westphal
Institute for Defense Analyses
1801 North Beauregard Street
Alexandria, Virginia 22311

Duc Tran
4923 Novak Lane
Fairfax, Virginia 22030

Abstract

The field of expert systems has claimed numerous and successful applications ranging from simple tasks, such as monitoring a valve or fluid rate, to very complex tasks that may include process scheduling or design. However, to construct a system requires the developer to isolate a set of rules that will guide the decision making process. Rules are conventionally defined during a series of interviews between a domain expert and a knowledge engineer. The encoding and representation of the extracted domain knowledge has proven to be a difficult barrier to overcome. KASH (Knowledge Acquisition SHell) has been designed to address this problem by providing a knowledge engineer with a set of utilities for constructing knowledge acquisition sessions based on interviewing techniques. The information elicited from domain experts during the sessions is guided by a question dependency graph (QDG). The QDG, defined by the knowledge engineer, is a series of control questions about the domain that are used to organize the cognitive processes of an expert. The content information supplied by the expert, in response to the questions, is represented in the form of a concept map. These maps can be constructed in a top-down or bottom-up manner by the QDG and used by KASH to generate the rules for a large *class of expert system domains.*

Introduction

Knowledge acquisition tools use a variety of techniques to elicit from a domain expert the essential information for building an expert system. The application domains of these tools can be aligned into two categories; *analysis* (e.g., diagnosis, identification, interpretation) and *synthesis* (e.g., scheduling, planning, design). Analysis is a top-down process providing an examination of a set of goals that are decomposed into simpler and more basic elements and relationships. For example, FIX (Rodi, Pierce and Dalton, 1989) is a diagnostic system based on class attributes for representing fault detection and isolation. Synthesis is a bottom-up process supporting the composition or combination of given facts and observations that are used to construct a set of goals. For example, SALT (Marcus, 1988) is a tool based on a propose-and-revise method for designing elevator configurations.

The above classes of tools work well within their intended domains, but are not easily transferred to other applications. Consequently, the time and effort required for producing the systems must be reinvested for each instance. This will increase the costs to the customer and subsequently decrease reusability and portability of the tools across domain applications. To address the different categories (analysis or synthesis), a system must be able to work within the constraints set by each. Expert system shells currently provide the capacity to build top-down or bottom-up applications by supplying the developer with a finite collection of objects to represent the domains. The manner in which the objects are defined and related to other objects determines the degree of acceptability of the system. Therefore, if the focus of producing a knowledge acquisition system is placed on the structure, rather than on content, a more diverse selection of applications may be addressed.

One knowledge acquisition system providing a domain-specific structure, KLAMShell (Cochran, 1988), was developed to aid in the construction of knowledge bases for maintenance and troubleshooting. The shell elicits information via subgoal satisfaction in a depth first manner to retain a context focus on the knowledge structure. Each goal defined is decomposed (via "push" menus) into a series of subgoals. The bottom-most nodes are transformed (via "pop" menus) into actions, such as questions or instructions, to be used in the final system. This process continues until all goals have been specified. The system is, however, domain specific and the generality of its guidance is limited to only a *small* subset of the domain, thereby restricting its use.

Another system that may be classified as a general purpose knowledge acquisition shell is PROTEGE (Musen, 1989). It is a tool capable of generating other knowledge acquisition systems using planning entities, task level actions, and input data. The methodologies used by PROTEGE separate the modeling of a domain (i.e., control structure) from the application knowledge (Musen, 1988), thereby customizing each system. However PROTEGE relies on a set of fixed templates to elicit information and this will limit the range of systems that can be produced.

From this perspective, KASH (Knowledge Acquisition SHell) has been defined as a domain-independent, knowledge elicitation shell. The shell provides a general purpose (reusable) environment for encoding the problem-solving methods of domain experts. A set of three independent modules (Figure 1) operate within the top-down (analysis) and bottom-up (synthesis) constraints of the domains. These modules are: *Concept Formulation*, for eliciting knowledge from a domain expert and structuring it into a concept map; *Knowledge Analysis*, to verify the concept map and cross check any inconsistencies found; *Rule Generation*, to produce a rule set for an expert system based on the concept map. A question dependency graph (QDG) supplies the control structures used in the concept formulation module. Control structures, defined *a priori* by knowledge engineers, are necessary to obtain the content knowledge from the experts. The utility of the QDG allows the control structures to be defined as needed and variations of the graph can easily address new applications.

Concepts and Base Facts

The basis for acquiring knowledge in KASH is organizing the cognitive processes an expert many formulate about a domain. The mechanisms to organize the processes must be tailored to the idiosyncratic representations created by the experts. However, experts tend to express their knowledge in unstructured and nondeterministic formats and difficulties arise when the knowledge must be converted into a reliable and useable format for expert system development. To properly address these representation problems, experts must be allowed, with minimal constraints, to describe their cognitive processes with respect to the intended application. Typically, the experts

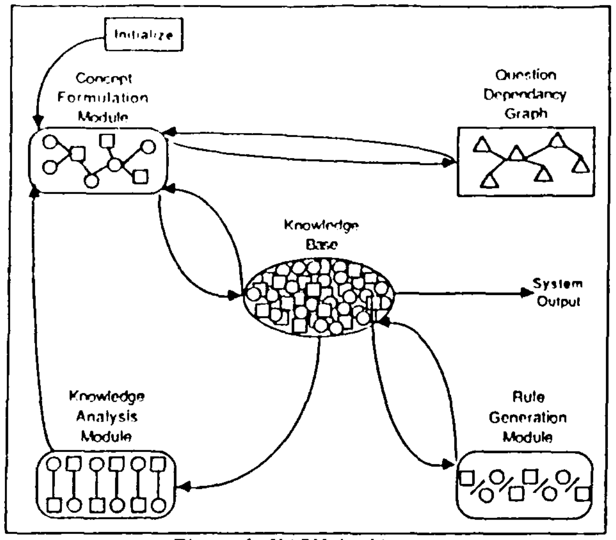

Initialize
Concept Formulation Module
Question Dependency Graph
Knowledge Base
System Output
Knowledge Analysis Module
Rule Generation Module

Figure 1: KASH Architecture

will define concepts that represent personal observations in the domain to be modeled. Concepts are defined as symbols (text or image) that capture the meaning or intention of objects and events of an environment (Ausubel, Novak and Hanesian, 1978). For example, an electrical system is an object, and boil water in an event. In essence, a concept is a *simplified* and *generalized* description of reality for a particular level of abstraction (Novak and Gowin, 1984).

Concept formulation may be defined as the process of extracting the characteristic features, termed criterial attributes, of the objects or events. The criterial attributes are what uniquely distinguish each concept (e.g., size, color, shape). A set of common features and the degree to which the features are accepted by the experts will determine the regularity (i.e., meaning) of a concept. The regularity implies precision on the definition of the concept so that misunderstandings can be minimized, see (Novak and Gowin, 1984). Criterial attributes also allow concepts to be grouped or linked together to form concept maps that can represent the conceptual and structural knowledge of the domain experts. A concept map is simply a hierarchical taxonomy of concepts. The hierarchy supports the natural subsumption of concepts where broad concept definitions are depicted at the top of the map and more detailed and concise concepts at the bottom. Concept maps promote the reuse of concept definitions by allowing the use of existing concepts (e.g., door) in different contexts (e.g., house, car). Such a virtual feature is natural for multiexpert integration because it provides support for a modular structure and the development of a library facility of reusable concept definitions across domains where applicable (McGraw and Westphal, 1988). The graphical nature of concept maps also allow multiexpert conflict to be made explicit thereby excising the knowledge structure of faulty linkages and misconceptions (Westphal and Reeker, 1990).

The concept map in KASH has been extended to include base facts. Base facts have their own definition because they are fully instantiated (e.g., parts, ingredients, symptoms) and do not require further justification for their existence. Base facts are measurable and observable in the context of the concept and support the criterial attributes of a domain. The criterial attributes are necessary for distinguishing between concepts and structuring questions into categories during concept formulation. The criterial attributes also support the rule generation module because they are variable entities (e.g., size is big, color is red) that can be compared, contrasted, or combined with other criterial attributes to form the rules produced by KASH.

Question Dependency Graph

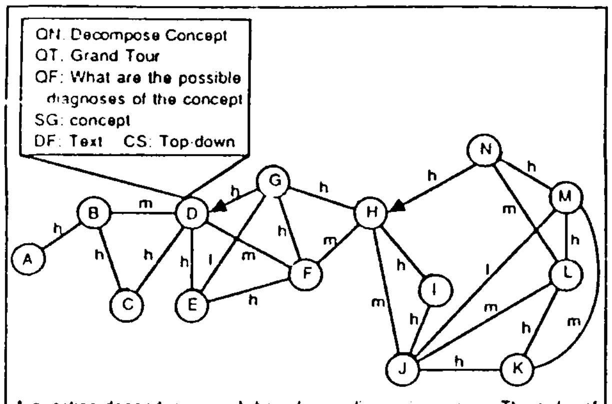
To facilitate the development of a concept map, the experts must be interviewed using questions pertinent to the content and structure of the problem domain. As Novak (1989) stipulates, the sequencing of questions presented to the expert should be tailored or grouped into specific sets of knowledge and the range of these sets should address questions at broad superordinate levels and become more narrow and precisely defined at the base level. In KASH, the questions permissible to ask of the domain experts during the development of the concept map are specified by the knowledge engineers and represented in a question dependency graph (QDG). The QDG has been abstractly based on the six types of questions as defined by LaFrance (1987) where *each question type decomposes the QDG into a category* of queries (broad and specific) to be asked with respect to conceptual definition. The six types are:

- *Grand Tour*, to identify domain and subdomain boundaries, e.g., "What is the purpose of this system?"
- *Cataloging the Categories*, to support, organize, and structure the concepts, e.g., "Can the concepts be ordered?"
- *Ascertaining the Attributes*, to specify values and ranges of the criterial attributes for concepts and base facts, e.g., "What values can attribute assume?"
- *Determining the Interconnects*, to define the causal relationships between concepts, e.g., "Does base fact support the concept?"
- *Seeking Advice* and *Cross Checking*, to compare and contrast information in the concept map, e.g., "Does concept contain any base facts?" or "What value of attribute is out of range?"

During the development of a QDG, a knowledge engineer must specify the questions that will be presented to the domain expert. The questions are defined to separate the control knowledge from content knowledge of the domain (analysis or synthesis). It is important for the knowledge engineer to understand the basic structure of the domain to effectively define questions, categorize the questions by type, and determine the control sequence of the questions. Inefficient specifications at this phase can lead to poor interview sessions with the domain expert, who will supply the content knowledge (response to the questions). In the QDG, the questions are represented as node structures and are linked to other node structures to determine the sequencing.

The node structure consists of a question frame, question type, selection guide, and a variety of support attributes. The question frame contains the actual text generated during a query. Questions are developed by the knowledge engineers about the control structure of the domain. The questions derived are used to respectively refine or compose the concepts in an analysis or synthesis domain. The six different question types defined above are used to insure the scope of the questions are limited to the current query category. By doing so, the focus of the experts are concentrated on the particular task level being modeled. The selection guide is satisfied when the knowledge engineer specifies the response expected from the question frame with respect to one (or several) of the system objects specified in KASH. These objects are a prioritized taxonomy of concepts, base facts, edges, attributes, and standard representation mechanisms (i.e, integers, reals, text, etc) that help to resolve which question to ask.

The edges form the explicit relationships between the node structures and determine which tuples of questions may be presented to the domain experts. A tuple of two questions is interpreted to imply: if question one is asked, then question two may be asked. There can be many-many relationships between the node structures, thus forming a large combination of questions tuples. The edges are coupled with edge weights to determine the ordering of question preference - a form of prefiltering. These weights may be displayed as a range of numbers or as an alternative representation (i.e., low, med, high).



A question dependency graph based on a diagnosis process. The order of node letters and the high, medium, or low priorities indicates direction. -> is used to denote explicit control.

Figure 2: Question Dependency Graph

Figure 2 shows an example of a partial question dependency graph for a circuit fault diagnosis. Each circle represents a unique question (defined by a knowledge engineer) that may be asked during a knowledge acquisition session. The box represents the node structure (only one is detailed) for a particular query scheme. The question name is Decompose Concept, the question type (QT) is Grand Tour, the question frame (QF) is defined as "What are the possible diagnoses of the concept?" (variable terms in the question will get instantiated during execution), the selection guide (SG) is expecting the response of the domain expert to be a concept, and a miscellaneous collection of support attributes helps to specify display formats (DF) and the control strategy (CS). The links between the nodes are labeled with high, medium, and low priorities to indicate the control paths the system should pursue. In this example node D is connected to nodes E and F. The heuristics defined in the concept formulation module will select the better candidate question (E or F) to ask based on the state of the concept map.

A QDG editor is used for the construction of the question graphs. The editor allows the knowledge engineers to construct specialized question bases through QDG link specifications. New questions may be introduced into the graph at any time. The set of base questions is always available, however it is the format in which they are connected that will give a system its functionality. This combination of nodes and links allow KASH to be used across a large range of applications and proves to be very versatile and powerful throughout the knowledge modeling process. As the QDG changes, so will the system, because the QDG is the canonical guidance referenced throughout the three modules of KASH: concept formulation, knowledge analysis, and rule generation.

Concept Formulation

Concept formulation, necessary to acquire the knowledge structure from a domain expert, is interleaved between a top-down and bottom-up elicitation strategy guided by the QDG. The top-down refinements are focused on acquiring the abstract concepts that are used to define the components of a domain. These components are in turn supported by bottom-up facts that represent the logical entities or extension of the concepts. The use of top-down and bottom-up strategies in concept formulation reflect the structures that exists in the different levels of knowledge and the control nature of the analysis and synthesis classes of problems[1]. The analysis problems involve identifying sets on objects based on their features. Synthesis problems construct a solution from component pieces or subproblem solutions. Figure 3 is a section of the top-down (and bottom-up) process trace of a concept formulation based on the QDG for a circuit fault diagnosis. The index (A-M) for each instantiated question text shown (i.e., variables are bound within the context of the question) corresponds to a node in the QDG shown in Figure 2.

A)Define the terminology:
Top-down terms(s) :> diagnosis.
Bottom-up terms(s) :> symptoms.

B)What is the purpose of
this diagnosis?
:>circuit fault.

C)Define any symptoms of
a circuit fault.
:>manifold threads.

D)What are the possible diagnoses
of a circuit fault?
:> control circuit failure.
:> sensor fault.

E)Can these be ordered?
:> no.

F)Is manifold threads a symptom of
control circuit failure, sensor fault?
:> sensor fault.

G)Are there any other symptoms
of sensor fault?
:> sensor mounts.

D)What are the possible diagnoses
of sensor fault?
:> broken sensor.
:> sensor grounding.

E)Can these be ordered?
:> no.

F)Is manifold threads a symptom of
sensor grounding, broken sensor?
:> sensor grounding.

F)Is sensor mounts a symptom of
sensor grounding, broken sensor?
:> sensor grounding.

G)Are there any other symptoms
of sensor grounding?
:> voltage test.

D)What are the possible diagnoses
of sensor grounding?
:> none.

D)What are the possible diagnoses
of broken sensor?
:> none.

H)What are some attributes
of sensor mounts?
:> connectors.

I)What value(s) can connectors
of sensor mounts assume?
:> loose, broken, grounded, working.

J)Is connectors used in
sensor grounding?
:> yes

K)What value(s) of connectors would
make sensor grounding succeed?
:> grounded, working.

L)What value(s) of connectors would
make sensor grounding fail?
:> loose, broken.

M)Is this a default or inferred value?
:> inferred

N)Can connectors be related
to other attributes?
:> no.

H)What are some attributes of
manifold threads?
:> connectors.

I)What value(s) can connectors
of manifold threads assume?
:> dirty, corroded, clean, working.

J)Is connectors used in
sensor grounding?
:> yes

K)What value(s) of connectors would
make sensor grounding succeed?
:> clean, working.

L)What value(s) of connectors would
make sensor grounding fail?
:> dirty, corroded.

M)Is this a default or inferred value?
:> inferred.

N)Can connectors be related to
other attributes?
:> no
•
•
•

Figure 3: Question Trace from QDG

The concept formulation module will iteratively execute six stages during the construction of the concept map. These stages are responsible for selecting a concept from the graph, establishing base facts for the concept, formulating a question to apply to the concept, generating the question, then presenting and accepting the results from the expert.

[1]Ausubel (1978) addresses these abstractions in his propositional learning theory.

Heuristic Criteria for Concept Selection - A concept is selected from the set of open concepts. This set represents those concepts that have not been fully explored by the QDG such that there are additional questions that may be asked about them. The concept is selected based on several factors including the order of importance that is explicitly specified by the user, a system defined depth first or breadth first selection mode, the recency of a concept definition, the level of attributes defined, the number of established links to the base facts, the number of derived (children) concepts, or a concept of enumerated type.

Establish Base Facts - The first step taken after the concept has been selected is to elicit the base facts that support the concept. These base facts define the fundamental units of the domain. The new base facts are specified by the user and supportive links are established to the selected concept. User selected base facts are then passed down to the selected concept from a parent concept. Note, parent concepts must resolve all supportive links before the concept map is considered complete.

Select Questions - The list of applicable questions for the current concept is calculated by obtaining all the edges attached to the previous question in the QDG and placing them into a list. A duplication filter is applied to the list to resolve any conflicts found in a list of previously ask questions maintained as a concatenation of the concept name and the question identifier. The list is then processed through some ordering filters that act on the edge weights, selection guide matching, and groups division types. The product of these filters results in a refined list of questions to be asked.

Generate Question - The product of the generate question stage is the selection of a single query node made from the refined list. This list of applicable questions for the current concept is acted on by a function of the question types and other system factors (not detailed here).

Question Presentation - After a single question has been selected it is presented to the user to solicit an answer. The control panel is consulted for the appropriate terms (top down/bottom up), display formats (bar graphs, text, pie chart, menu), and concept graph information (attributes, types, links).

Answer Questions - The expert responds to the question through either text (e.g., attributes, concepts, base facts, names, or values) or concept graph management facilities (e.g. graphics, node selection).

The output expected from the concept formulation module is a concept map representing a top-down or bottom-up process model. The concept map represents the observations of domain experts as guided by the control structure specified in the QDG. Figure 4 shows the concept map for the circuit fault example. If the execution of the module has not fully defined the range of the expert's knowledge, the QDG may be reconfigured to accommodate the missing information. From this point the information gleaned from the expert in the form of concept modules and base facts is verified by both the system and the domain expert. Several structure analysis techniques, described below, have been defined to accommodate this task.

Knowledge Analysis

The knowledge analysis module is a hybrid of several interactive subsystems that are used to validate and refine the concept map. During validation, the knowledge acquired from the concept formulation module is analyzed to bring out inconsistent, incomplete, and unjustified information that may occur in particularly large systems or from the use of multiple experts. In refining the concept map, the system will look for ways to enhance the structure of the model through the application of a variety of techniques. Several proposed techniques are described below, also see Figure 5.

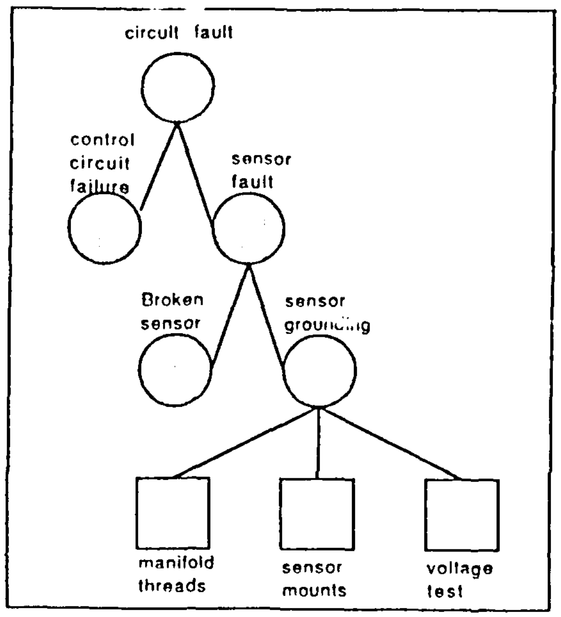


Figure 4: Concept Map for Circuit Fault

Cluster Analysis - Cluster analysis detects cases in which large numbers of concepts are derived from (or clustered around) a more general concept. It suggests that the general concept is too broad and it should imply several new concepts where the other concepts can be derived from, or some of the derived concepts should be promoted to the the intermediate level concepts between the general concept and others. The introduction of intermediate concepts helps structure the knowledge elicitation stage and reduces the complexity of the acquired knowledge.

Entailment Analysis - The basis for entailment analysis stems from the attributes associated with the concepts. The technique makes the entailments between concepts obvious. There will be cases in which the entailments indicate that a concept can be subsumed by another concept because it is derived by another concept that has no other derivation.

Similarity Analysis - The knowledge engineer submits to the system a number of known cases for analysis. Each of these cases is represented in the format compatible with the acquired knowledge. The system will try to justify the given facts and results of the cases. If the facts and results cannot be explained, the system will explain the failure and it will seek help from the user by asking questions from the QDG focusing on the unexplained concepts and the failure.

Relative Analysis - Relative analysis is based on the reuse of concept modules in the graph. A concept associated with a base fact is selected for review. The analysis technique will propagate up the structure to a level outside the immediate hierarchy associated with the concept and query the expert, via the QDG, if it can be used in the adjacent paths. This type of analysis supports the use of virtual structures and serves as a form of memory cue entailment for additional structure refinements.

Subcomponent Analysis - A base fact is selected from an active path in the concept structure. This path consists of the base fact and all the concepts that comprise the path to the root node. The base fact is compared to each the concepts that subscribe to the path and questioned by the QDG for relevancy against the concept. Negative responses will indicate inconsistencies in the structure and will be resolved through further QDG examinations.

Conceptual Analysis - Attempts to enforce the principal that every base fact must be attached to a concept that is not further decomposed by additional concepts. If any such case is encountered it can be assumed that there are additional conceptual levels that have not been defined. Additionally, if a concept has been decomposed into another single concept, then the linear formation will invoke the analysis to collapse both into a unary concept structure.

Knowledge analysis can be interactive or selectable. In the first mode, *interactive*, a number of knowledge analysis techniques are selected from a menu before the concept formulation module is executed. As knowledge is acquired these techniques are automatically invoked to yield analysis results. The results are shown in a side window with the appropriate marks to indicate the different degrees of importance. In the second mode, *selection*, the knowledge analysis stage is entered after a knowledge elicitation session. The manner in which the analysis results are shown is the same as in the previous mode, but more thorough analysis techniques can be applied, particularly the similarity analysis. The topics of refinement and analyses to be performed are based on a prioritized scheme and will execute after any data changes or alterations to the concept structure are made.

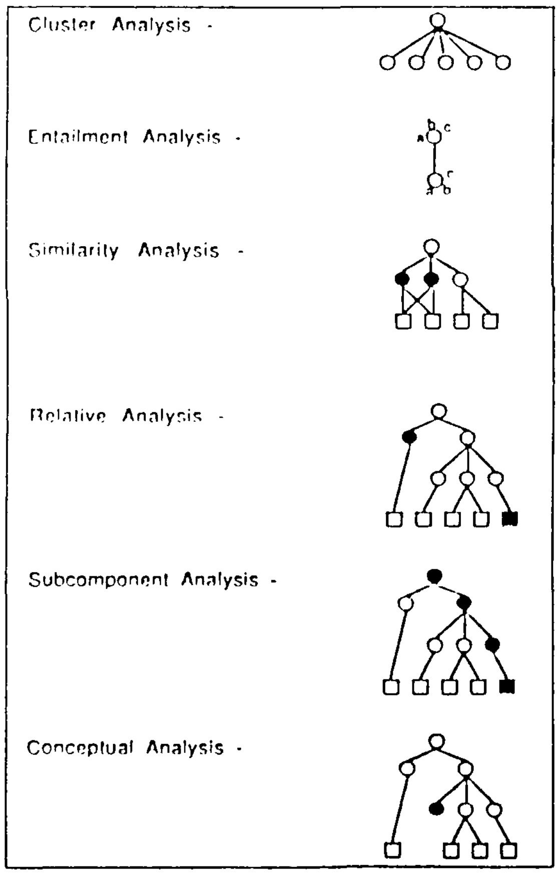


Figure 5: Knowledge Analysis Techniques

Rule Generation

The third module of KASH is used to support the final process of knowledge acquisition, rule generation. This module creates a set of rules that can be integrated into a third party expert system shell. The three major factors that influence the generation of the rules are the concept map, rule generation strategy, and target expert system shell.

First, the concept map is extended to include any terms (criterial attributes) to be used in the production of the rules. New attributes are created and manipulated by *attribute<->attribute*, *attribute<->concept*, and *attribute<->value* questioning schemes defined in the QDG. The information provided by the experts in response to the questions specifies the attributes of a concept, the relationships of the attributes to other attributes, and the range of values the attributes may assume. Attribute definitions are initiated for each of the base facts in the concept map. The definitions are propagated to the concepts via an inverted inheritance scheme with question filters to terminate any attribute stream. The attribute relationships in the concepts represent rules. The rules are local to the concept for which they are defined and may be classified as either *default*, where new logical relationships between attributes are specified (i.e. set the temperature of material to equal the temperature of the gas), or *inferred* where the system must calculate the value of the used attribute (i.e., the temperature of the material must equal the temperature of the gas). The inferred values are present in the rule premises and the default values typically appear in the consequents.

Second, in the rule generation strategy the local rules are used by a series of high level top-down and bottom-up strategies that determine the direction in which final rules are developed. The top-down strategy employed in generating rules is based on a concept and the links to its supportive concepts. The bottom-up strategy directs the rule generation based on the lower level concepts to the higher level concepts that they support. This strategy supports both forward and backward chaining rule specifications for integration into the expert system.

Third, the characteristic or model of the target expert system shell language influences the different ways in which an optimal rule set may be generated. This depends on the different representation schemes, control strategies, deamons, triggers, etc. encountered in the different expert system shells. For example, instead of explicitly stating separate rules as would be necessary in a pure production system such as OPS-5, the number of rules generated can be reduced in an object oriented system such as NEXPERT or KEE due to the abstraction of frame structures.

Figure 6 shows the partial set of rules that have been created for the circuit fault diagnosis example. Before these rules are converted into a target expert system format they must be reviewed by a series of analysis techniques to ensure there are no redundant, conflicting, or circular rules. The final outcome is a knowledge base of objects, relationships, and rules that may be used in a variety of expert system applications.

Supportive Modules

There exists a number of external features that will be added to the basic functionality of KASH. These features are intended to extend support to the knowledge acquisition process and supplement the work of the QDG. The following represent a subset of the planned features to be developed:

• The time issues associated with a domain model will be addressed through a temporal subsystem. The concept graphs will be modified to accommodate the values necessary to represent the start, end, and latency periods. The question dependency graph will also be updated to elicit the necessary information. The temporal information will be represented through a *time box* consisting of a series of levels each partitioned into the respective intervals specified during the temporal questioning. This time box can be viewed from varying levels of detail depending on the concept graph depth.

• An enhanced control panel will contain a multitude of user defined parameters to coordinate the elicitation sessions. The panel will contain the terms used for the top-down and bottom-up definitions. Pre-logical settings for which top-down entries are best suited for the bottom-up selections will be available. The search strategies (i.e., depth first, breadth first) can be set from the panel. A question list is defined in the panel and used to disable groups of questions for the session (usually low priority settings). Additionally, the analysis techniques can be made interactive or selectable from the panel settings.

• A history window will supply a meta-command interpretation of the events occurring in the system. This will provide documentational support to analyze the elicitation session as it progresses over time. The log will be used to archive the state of the concept model for easy reconstruction at a future date. Additionally the history list will be constructed for path resolution of the virtual system hierarchy.

• A glossary of terms defined by the expert during the elicitation of concepts and facts will be supported as a scrolling series of menus. This facility can support the comments associated with multiple expert development. The glossary support system can be expected to emulate a hyper-expert notation scheme where the terms defined can be a mixture of text and graphics.

```
IF level_of_voltage_test < 3
THEN position_of_sensor_grounding = "open"

IF level_of_voltage_test >= 3
AND level_of_voltage_test < 5
THEN position_of_sensor_grounding = "closed"

IF connectors_of_sensor_mounts = "loose"
THEN cond_of_sensor_grounding = "poor"
AND recommendation_of_sensor_grounding = "tighten"

IF connectors_of_sensor_mounts = "broken"
THEN cond_of_sensor_grounding = "poor"
AND recommendation_of_sensor_grounding = "replace"

IF connectors_of_manifold_threads = "dirty"
THEN cond_of_sensor_grounding = "poor"
AND recommendation_of_sensor_grounding = "clean"

IF connectors_of_manifold_threads = "corroded"
THEN cond_of_sensor_grounding = "poor"
AND recommendation_of_sensor_grounding = "replace"

IF cond_of_sensor_grounding = "poor"
AND position_of_sensor_grounding = "open"
THEN status_of_sensor_fault = "true"
AND recommendation_of_sensor_grounding =
        recommendation_of_sensor_fault
AND action_of_sensor_fault = "shutdown"

IF cond_of_sensor_grounding = "poor"
AND position_of_sensor_grounding = "closed"
THEN status_of_sensor_fault = "true"
AND recommendation_of_sensor_grounding =
        recommendation_of_sensor_fault
AND action_of_sensor_fault = "maintenance"

IF status_of_sensor_fault = "true"
THEN status_of_circuit_fault = "true"
AND recommendation_of_circuit_fault =
        recommendation_of_sensor_fault
AND action_of_circuit_fault =
        action_of_sensor_fault
```

Figure 6: KASH Circuit Fault Rule Set

Conclusion

KASH is a general purpose knowledge acquisition shell that acquires information about a domain from an expert. Because KASH has been designed to support both analysis and synthesis applications, it may be applied to a broad range of systems including planning, design, classification, scheduling, decision support, and diagnosis where complex knowledge is often acquired from multiple domain experts. The three modules supplied by KASH provide the capability to acquire knowledge in context (i.e., customized for each domain): *Concept Formulation*, for structuring and eliciting the knowledge from a domain expert into a concept map; *Knowledge Analysis*, to validate the concept map; and *Rule Generation*, to produce a rule set for an expert system based on the concept map. All the information elicited from a domain expert is guided by a question dependency graph (QDG). The QDG is developed by a knowledge engineer to separate the control knowledge from the application knowledge. Thus, the QDG is reconfigurable for customerization across applications and domain experts. Furthermore, KASH has the capabilities for producing a knowledge base that can be used to generate alternative rule sets for different expert system shells.

Acknowledgements

The authors would like to give special thanks to Joseph Novak for discussing his experience with concept maps; Ken Ford and Jack Adams-Webber for their support and ideas; Dick Wexelblat, Karen McGraw, and Mike Bloom for their initial reviews. The views and conclusions contained in the paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of their respective companies or governmental institutions.

References

Ausubel, D.P., J.D. Novak, and H. Hanesian (1978). *Educating Psychology: A Cognitive View*, 2nd edition, New York: Holt Rinehart, and Winston.

Cochran, E.L. (1988). "KLAMShell: A domain-specific knowledge acquisition shell," Technical Report CSDD-889-16301-1, Honeywell, Golden Valley, MN.

LaFrance, M. (1987). "The knowledge acquisition grid: A method for training knowledge engineers," *International Journal of Man Machine Studies*, 26, 245-255.

Marcus, S. (1988). "SALT: A knowledge-acquisition tool for propose-and-revise systems," In Marcus, S. (ed), *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers,Boston, pp. 81-122.

McGraw, K.L. and C.R. Westphal (1988). "Accommodating multiple expert input during knowledge acquisition: Integrating hypertext with a knowledge acquisition tool," *Sixth Annual Intelligence Community AI Symposium*, Langley, VA.

Musen, M.A. (1989). "Knowledge acquisition at the metalevel: Creation of custom-tailored knowledge acquisition tools," *SIGART Newsletter*, No. 108, April. pp. 45-55.

Musen, M.A. (1988). "Conceptual models of interactive knowledge acquisition tools," *Proceedings of the European Knowledge Acquisition Workshop*, Bonn, West Germany.

Novak, J.D. (1989). "Helping students learn how to learn: A view from a teacher-researcher," *Proceedings of the Third Congress on Research and Teaching of Science and Mathematics*, Santiago de Compostela, Spain.

Novak, J.D. and B.D Gowin (1984). *Learning How to Learn*, New York: Cambridge University Press.

Rodi, L.L., J.A. Pierce, and R.E. Dalton (1989). "Putting the expert in charge: Graphical knowledge acquisition for fault diagnosis and repair," *SIGART Newsletter*, No. 108, April.

Westphal C.R. and L.H. Reeker (1990). "Reasoning and representation mechanisms for mutltiple-expert knowledge acquisition," in *Knowledge Acquisition: Current Practices and Trends*, McGraw, K.L. and C.R. Westphal (eds), Ellis Horwood, Chichester, England.